Girls' Programming Network

Guess Who!

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney and Canberra

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Caitlin Macleod
Rachael Newitt
Imaina Widago
Alesiya Maynard
Courtney Ross
Jeannette Tran
Fiona Lin
Renee Noble
Alex McCulloch

Testers

Sheree Pudney
Claire Quinlan

Part 0: Getting set up!

Task 0.1: Set Up the File

Create a file where we are going to write the code for our game.

1) In your Python 3 IDLE click ***File*** and create a ***New File***.
2) Start by saving your file and calling it ***guess.py***

Now when you want to run your code, just click ***Run*** (or press ***F5*** on the keyboard)!

Great! Now we're ready to code!

Task 0.2: You've got a blank space, so write your name!

At the top of the file use a comment to write your name!
Any line starting with `#` is a comment.

```
# This is a comment
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

☐ You should have a file called guess.py

☐ Your file has your name at the top in a comment

☐ Run your file with F5 key and it does nothing!!

Part 1: Welcome Message

Task 1.1: Print a welcome message

We want to `print` a message to tell the user what our program does.

1. On the line after your name, use the `print` statement to display the following message:

```
------------------------------------------------------
Welcome to Guess Who!
------------------------------------------------------
Moves: Pick a person, and let the computer guess who
you're thinking of. Type "yes" or "no" to answer the
questions.

Good luck!
```

Don't want to type all this out? Go to http://bit.ly/gpn-2018-4.

Hint

Want to `print` multiple lines at a time? You can use three sets of quotes instead of one, to make your strings go over multiple lines

```
print("""
Print
Three
Lines
""")
```

Task 1.2: Copy in the list of people

We need to create the list of all the people in our Guess Who game! This list will also contain a list of all their attributes.

1. Copy and paste the list from http://bit.ly/gpn-2018-4 and assign it to a variable called **`people`**.

2. Format the list of lists by going to top menu bar, click *`Format -> Format Paragraph`*. This will make the list of **`people`** easier for us to read.

Task 1.3: Hide that character!

In this game, we'll be getting the computer to guess who the user is thinking of! Select a person from the list of people to hide. This workbook will refer to that person as the `hidden character`.

★ Bonus 1.4: Who do you know? ★

Add additional people to your list! They need to each have an eye colour, hair colour, and accessory.

You can add as many as you like, but make sure no one has exactly the same hair, accessories and eye colour as someone else!

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

- ☐ Print a welcome message
- ☐ You have a list of people.
- ☐ You have selected a character for the computer to guess.
- ☐ Run your code!

Part 2: Selecting Attributes!

Task 2.1: What's your style?

Our people all have different attributes, in the order of name, eye colour, hair colour, and then accessory. We need to create lists of all the different options!

1. Create a list of all the different possible eye colours, and store it in a variable called `eye_colours`.
2. Do the same thing for all the different hair colours, and then the accessories! Call the variables `hair_colours` and `accessories`.

Make sure that each option that you included in `people` is also stored in the lists above!

Task 2.2: Creating looks

So the computer can start guessing, we need the computer to select an option from each of our `eye_colours`, `hair_colours` and `accessories` lists!

1. Select an item from `eye_colours`. Store it in a variable called `eye_guess`.
2. Do the same thing for `hair_colours` and `accessories`. Call the variables `hair_guess` and `accessory_guess`.

Hint

We can access items in a list individually. The below code will print out the second item in the people list:

```
print(people[1])
```

Don't forget that lists start from 0!

Task 2.3: Do they look like this?

The computer needs to find out if the eye colour, hair colour and accessory they selected match the eye colour, hair colour and accessory of the `hidden character`.

1. For the `eye_guess`, use `input` to ask the user if it matches the eye colour of the `hidden character`. Store the answer in a variable called `eye_guess_answer`.
2. Do the same thing for `hair_guess` and `accessory_guess`. Store the answer in variables called `hair_guess_answer` and `accessory_guess_answer`.

★Bonus 2.4: Uppercase or lowercase

Sometimes users don't type exactly what we expect them to! If you're expecting a user to type `"yes"` or `"no"` but they type `"Yes"`, `"YES"` or `"NO"` your code may not recognise their answer correctly.

Make your game recognise user `input` if they enter versions of your expected input with different capitalisation.

Hint

`"FrOg".lower()` will return `"frog"`. Try use `.lower()` on your variables to make sure the human players move is converted to lowercase!

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

☐ You have a list called eye_colours

☐ You have a list called hair_colours

☐ You have a list called accessories

☐ The computer has selected an eye colour, hair colour and accessory to guess.

☐ The computer asks the user if their hidden character has the eye colour, hair colour and accessory that the computer picked and stored the answers.

Part 3: Narrowing it down!

Task 3.1: Splitting out people!

Now that we know whether the `hidden character` has the attributes the computer guessed or not, we need to compare it to the list of `people`.

1. Select the first person in the list of `people`. Store it in a variable called `person`.

Task 3.2: Splitting out attributes

For each person, we need to check their eye colour, hair colour, and accessory!

1. For the `person`, get their name. Store it in a variable called `person_name`.
2. For the `person`, also get their eye colour, hair colour and accessory. Store it in variables called `person_eye`, `person_hair` and `person_accessory`.

Task 3.3: Manual Deletion!

Let's try seeing how people will be eliminated, and what our people list will look like after we've eliminated everyone with brown hair. Cross off anyone who has brown hair! Don't forget to include any characters you added.

 Aleisha	 Brittany	 Charlie	 Dave	 Eve	 Frankie
 George	 Hannah	 Isla	 Jackie	 Kevin	 Luka

Task 3.4: Do they match?

Now that we have the person's attributes, and the user has answered the computer's eye colour, hair colour and accessory guesses, it's time to work out if we can eliminate anyone!

What are the options for eye colour? Let's assume that the `hidden character` has blue eyes:

Guess	Yes	No
Blue eyes	Keep	Eliminate
Not Blue eyes	Eliminate	Keep

If the computer guessed that the `hidden character` has blue eyes, and the user answered `"yes"`, then any character that doesn't have blue eyes needs to be eliminated. Otherwise, if the user answered `"no"`, then any character that does have blue eyes needs to be eliminated.

Can you fill out this table for hair colour? Let's assume that the hair colour of the `hidden character` is brown:

Guess	No	Yes
Brown Hair		
Not Brown Hair		

Let's do it one more time, this time for accessory! Let's assume that the `hidden character` has no accessory:

Guess	Yes	No
Has accessory		
Has no accessory		

Now we know all the ways that a person can be eliminated!

Task 3.5: What if?

Now that we know all the different ways that a person can be eliminated, we can code it using `if` and `elif` statements!

1. Create `if` and `elif` statements to check the eye colour. If the `eye_guess` was correct and the `person_eye` does not match `eye_guess`, `print` out `"Eye colour does not match!"`. If the `eye_guess` was wrong, and the `person_eye` does match the `eye_guess`, also `print` out `"Eye colour does not match"`.

2. Do the same thing as you did in step 1, but for checking the hair colour! Make sure it's part of the same `if-elif` chain by continuing with `elifs`!

3. Do the same thing you did in step 1, but for checking the accessory! Make sure it's part of the same `if-elif` chain by continuing with `elifs`!

Hint

In `if` statements, we can use the keyword `and` to check if multiple things are true:

```
if raining == True and umbrella == "I forgot it!":
      print ("Don't go outside!")
elif raining == False and umbrella == "I forgot it!":
      print("It's okay, it's not raining")
elif raining == True and umbrella == "I've got it!":
      print("Awesome! Let's go outside!")
```

Hint

Why so many elifs???

We need to use and `if-elif-elif-elif-elif-elif` chain because we only want to add the character to the elimination list once! If we use several `if-elif` pairs then we might add the character to the elimination list of multiple times for different features!

If we try and eliminate them multiple times the computer will be confused because they are already eliminated.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

☐ Get all the attributes of the person

☐ Compare all the ways a person can be eliminated

☐ If-elif statements list all the ways that a person can be eliminated, and print out when they are

☐ Try running your code!

Part 4: Eliminate! Eliminate!

Task 4.1: Again, Again, and Again!

Now that we've checked to see if the attributes of one person matches what the computer guessed, we want to be able to check everyone in the people list! To do this, we're going to use a `for` loop.

1. Use a `for` loop to go through each person in the `people` list to check to see if they need to be eliminated.

2. Make sure that all the code from section 3 is inside the `for` loop!

Hint

Indented lines have a tab at the start like this, they look this:

```
for blah in something:
    THIS IS INDENTED
```

Task 4.2: Make a list of things to eliminate

The computer needs to track all of the people that it knows isn't the correct answer. We're going to store this in a separate `list` for now.

1. Create an empty `list` and assign it to a variable called `eliminate`.

Task 4.3: Make a list of things to eliminate

We need to add all the people that need to be eliminated to the `eliminate` list! In your `if` and `elif` statements that were created in section 3:

1. Every time there isn't a match, update your code so instead of printing something, we're going to add the `person` to the `eliminate` list.

Hint

You can add items to lists using the append statement:

```
dinner = []
dinner.append("pizza")
```

Task 4.4: Eliminate Them!

In another `for` loop, go through each person in the `eliminate` list and `remove` them from the `people` list. This way, the computer won't try to guess them.

1. Create a `for` loop that goes through each person in the `eliminate` list.
2. Remove each person from the list of `people`.

Hint

If I wanted to remove an element from a list I could use code like this:

```
dinner_options.remove("pizza")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:

☐ Your code loops over every person in the people list

☐ Your code removes people already identified as eliminated from the list of available people.

☐ Try printing out your list of people before and after eliminating characters!

Part 5: Guess Them!

Task 5.1: Making the guess

It's time for the computer to guess who it is! The computer needs to ask the user if that's the `hidden character` by using person's name.

1. Pick the first person from the list of people not yet eliminated. Store it in a variable called `guess`.
2. From `guess`, get the name of the person. Store it in a variable called `guess_name`.
3. Use `input` to ask the user if the computer guessed the name correctly. Store the answer in a variable called `answer`.

Hint

Remember that `people` is actually a list of lists! You may find it useful to `print` out `guess` and `guess_name` to help check that you're accessing the list correctly.

Task 5.2: That's correct!

If the computer guessed the right person, it's time to celebrate! Get the computer to `print` out a message about how great the computer is at this game, and how lovely it was to play with the user.

1. Create an `if` statement that checks to see if the guess was correct. If it was, `print` out a congratulations and thank you message.

Task 5.3: Wrong answer!

If the computer didn't guess correctly, the person they guessed should be removed from the list of people so they don't get guessed again.

1. Update the `if` statement you created in task 5.2 to have an `else`.
2. In the `else` statement, remove the guess from the list of `people`.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

☐ The computer selects the first person from the list of people, and guesses who!

☐ The computer responds to a correct guess by printing a congratulations message!

☐ The computer responds to an incorrect guess by removing the character from the list of possible people.

Part 6: Randomize it!

Task 6.1: Import random library

It's really boring that our computer only guesses the same eye colour, hair colour and accessory! Let's randomise what the computer picks.

At the top of your file add this line:

```
import random
```

Task 6.2: Pick a random look

Now we need to update how the computer makes its guesses!

1. Update the code where the computer selects the `eye_colour_guess` so that it's randomly selected!
2. Do the same for `hair_colour_guess` and `accessory_guess`!
3. Now randomly select the `person` to `guess`!

Hint

If I wanted to choose a random food for dinner I could use code like this:

```
dinner = random.choice(["pizza", "chocolate", "nutella", "lemon"])
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 7:

- ☐ Import random
- ☐ Pick a random value from eye_colours
- ☐ Pick a random value from hair_colours
- ☐ Pick a random value from accessories
- ☐ Pick a random person to guess from people
- ☐ Try running your code!

Part 7: Break the loop

Task 7.1: Add the game loop!

Create a `while` loop that runs forever, so the computer can ask as many questions as it wants!
You'll need to use:

- A `while` loop
- A `True` statement

The `while` loop will run as long as what comes after the `while` is true. The easiest way to do this is using a boolean `True`.

Use this line to make the game play on repeat

```
while True:
```

Hint

You will need to indent all the code that you want looped!

```
while True:
    THIS IS INDENTED
```

Task 7.2: To infinity and beyond!

Whoops! It looks like we created an infinite loop - the game never ends! You can press `CTRL+C` to stop your program.

We need to `break` the loop!

1. Update the `if` statement that checks to see if the computer guess correctly to include a `break` statement.

★Bonus 7.3: Liar, Liar!

What if our user wasn't telling us the truth? If we get to the end of guessing and there's no more people to guess, what happens?

Run your code and see if you can make the computer run out of things to guess!

Hmmm….We should fix that. Let's add a check to check how many people are left in the list and yell at our user for trying to trick us if there is 0.

1. Create an `if` statement that checks the length of the `people` list.
2. If there is no one left, `print` out a message that says `"You're playing tricks on me! There's no one left :("`

Hint

You can check the length of the list using `len()`.

```
len(["pizza", "chocolate", "nutella", "lemon"])
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to the Extensions:

☐ Your code runs without any problems.

☐ Guessing the right person ends the game.

☐ When the game is over, you break out of the loop.

Part 8: Extension: Smarter guessing!

Task 8.1: Process of elimination

Let's make our computer smarter!

To do this, we're going to work through each of the lists for eye colour, hair and accessories from beginning to end.

For example, our hair colours are:

```
hair = ["black","brown","red"]
```

Imagine if we ask our user if the hair colour is black, and they say no.

Then we ask if it's brown, and our user says no.. what colour is the hair?

There's only one color left, so we know the hair colour must be red!

Let's change our code to get rid of `random.choice` and replace it with list indexes. Every time we get a hair colour from the list, we want to get the first option.

Go back and look at part 3 if you need a reminder about how list indexes work.

Task 8.2: We like to .remove() it, .remove() it!

Now we're getting the first item in the hair colour list every time. But, because our list is the same we always choose `"black"`.

To fix this we need to make sure we remove the bad items from the list, so we don't ask about it again.

Task 8.3: No questions asked

Just like the example before, if there's only one choice left, we don't need to ask whether it's the right one, we already know!

Change your code so that if there's only hair colour left in the list, we don't ask any more questions about hair colour.

Task 8.4: Off we go again!

Our hair colour guessing is excellent now, but we can definitely make the others better too.

Go back and improve the guessing about eye colour and accessories to make them better as well.

☑ CHECKPOINT ☑

If you can tick all of these off, you have finished this part:

☐ Your guessing for eye colour, hair colour and accessories all work using list indexes

☐ When a guess is wrong, you remove it from the list

☐ When the lists are only one element long, you don't ask any more questions about that characteristic

Part 9: Extension: Read it in!

Task 9.1: Where have all the people gone?

1. Create an empty list of `people`.

You can `comment` out your list of `people` from earlier or delete it, whichever you prefer.

Task 9.2: Here they are!

1. Download the file `guess_who_people.txt` from http://bit.ly/gpn-2018-4!
2. Make sure you save it in the same directory as your python file.

Task 9.3: Open sesame!

Use Python's *`with open`* to open the the text file.

Use this line just after you create your empty list to open your people file and read what it says.

```
with open("guess_who_people.txt") as f:
```

Task 9.4: Let's loop again

So we can open the file, but how do we get the people out?

We make another loop of course!

You'll need to use:

- A `for` loop

Use the code below inside your open statement to help you read in each of the lines in the file, one by one.

```
for line in f:
    line = line.strip()
    parts = line.split()
```

Hint

`with open` and the `for` loop both need to be indented. So if you're getting an error, make sure to check that your code is indented like below.

```
for blah in something:
    THIS IS INDENTED
    for loop in loop:
        THIS IS REALLY REALLY INDENTED
```

Task 9.5: Append your people!

Now we have each of the people in the file, we want to add them to our `people` list. Try to do this using `append()`.

☑ CHECKPOINT ☑

If you can tick all of these off, you have finished!

- ☐ You are using "with open" to open a file
- ☐ You use a loop to read each line in the file
- ☐ All of the people are appended to your people list
- ☐ Your code runs without any problems